ASSIGNMENT #4

5% of the final grade COEN 352 - Fall 2016

Due Date: 17.11.2016 by 11:59PM

Please manage your time so that you can finish all parts of this assignment on time.

All homework assignments will be posted on the course's Moodle website.
You must submit your homework using Moodle.

The objective of this assignment is to get you to practice advanced tree construction and utilization for the purpose of game playing. You need to build all tree structures and algorithms operating on them yourself.

The game of 'Walls' is comprised of an NxN board of squares (like chess, for example) and two players (say A and B), where N is odd and greater than 1. Each game is played twice, one time with player A starting the game and another with player B making the first move. The game has a *start state*, *play rules* that determine legal moves and an *end state*. Once a game ends, there is a certain accounting that decides the *game result*. These concepts are explained below. I suggest you first design a game with N=3.

Your objective, as the programmer is to design, in detail, build and test an *artificial agent* (AA for short) capable of playing the game, as optimally as possible, against a human player or another AA. To do so you need to build a *search tree* (often called game tree) that includes every possible move starting from the initial state of the game, and all possible future moves, all the way down to the leaf nodes (dead-ends with no possibility of future moves). The search tree is searched to decide on the best current move, which is the move that is most likely to lead to an eventual win in the game. An example of a search tree and how it is used to decide game is referred to below, also.

Start state. The game starts with a *clear* board with the two players placed at horizontally symmetric positions, which are at equal distances from and as close as possible to the *center* of the board (without overlapping). A player has a 'face' which indicates the *direction* of its forward movement. At the start of the game each player is facing *outward* towards the closest edge to it.

Play rules. The players take turns making moves. A player can: try and move forward (*try-forward*), turn 90 degrees left (*turn-left*) or turn 90 degrees right (*turn-right*). A player *can* only move into an empty square or into a square with a 'brick' bearing its name. A player *cannot* move into a square that has a 'brick' of the opposite type, or off the edge of the board, or into a square currently occupied by the opposite player.

Anytime player *X* moves forward from an unmarked square, that square is filled with a brick marked *X* (signifying its origin).

End state. The game ends when either every square on the board contains a brick or *both* players are *stalled*. A player is stalled, *by definition*, if it makes 9 consecutive moves without laying a new brick. A stalled player is not allowed to make any more moves.

Game result. Once the end state of the game is reached, the player with the greater number of bricks bearing its name wins 2 points (the other receiving 0). If both players lay the same number of bricks then the result is a draw, with each player receiving 1 point.

Search Trees. See:

http://www.cs.toronto.edu/~hojjat/384w09/Lectures/Lecture03-GameTreeSearch-4up.pdf
http://neverstopbuilding.com/minimax

Bonus Q. Everyone that completes this assignment, without error will receive a bonus.

Programming Language and IDE:

You can use Java or C++.
Use Eclipse for Java and Visual Studio for C++.
If you write you program on Mac Machine, make sure it runs on a Windows machine.
If the marker cannot run your program, you will lose at least half of the assignment's mark.

Continue

Documentation:

Metadata

- The programming assignment **title**—e.g., "Problem Set 4: SearchTrees."
- The name and netid of the **student** submitting the assignment.

Design and Implementation

None requested.

Testing

(a) **Screenshots** of results

How to Submit Assignment #4:

At the top of each assignment, include comments that consist of your name, your student ID and the assignment name or number.

When you complete your assignment #4, you will have Source code(s), Documentation and executable file:

Fristname_Lastname_A4.cpp (or .java)
Firstname_Lastname_A4.doc (or docx or pdf)
Firstname_Lastname_A4.exe (or jar)

(First name and Last name are your actual name, i.e. Nancy Nelson)

Here is the instruction on how to submit your assignments, in this case, Assignment #4:

- Create a folder on the desktop, name it: Your Name_A4 (i.e. Assignment #4 Nancy)
- Locate your assignment **A4 (source code, documentation, executable file)**, Copy and Paste them inside **Your name A4** folder
- Close the folder (if it is open)
- Right click on the folder, select "Send To" and then, click on the "Compressed (zipped) Folder"

Now, you should see another folder on the desktop with the "zip" extension: "Your Name_A4.zip"

Now you should be ready to upload your folder and submit it using the Moodle.

Continue

Honesty Policy: It is expected that all students will conduct themselves in an honest manner, and never claim work which is not their own. Violating this policy will result in a substantial grade penalty, and could result in expulsion from the University. However, students are allowed to discuss programming assignments with others and receive debugging help from others.

****

The same instruction applies to ALL assignments.

****

IMPORTANT:
You must give a proper name to your files prior to submitting them. Each file must start with your first name, and then, last name and Assignment Number. Here is an example: ***david_jones_A1.zip***

IMPORTANT:
You will be responsible to see that your homework is submitted correctly.

IMPORTANT:
If it has been submitted incorrectly, you can resubmit it. The system keeps track of the last file submitted for each program. If it is resubmitted after the due date, it will be considered late. The program automatically dates and times your program and identifies if it is late or not.

IMPORTANT:
Files submitted for homework may be electronically compared to detect cheating.

VERY IMPORTANT:

Each assignment is due **PRIOR** to 11:59PM. If an assignment is submitted at 12:00AM or after, even it may be 1 or 2 minutes late, it is considered a late assignment and will get zero! No Exception Can Be Made!
Always double-check before submitting your assignments.